Introduction

The Knowledge Systems Laboratory (KSL) is an artificial intelligence (AI) research laboratory of over 100 people—faculty, staff, and students—within the Departments of Computer Science and Medicine at Stanford University. KSL is the new name for the interdisciplinary AI research community that has evolved over the past two decades. Begun as the DENDRAL Project in 1965 and known as the Heuristic Programming Project from 1972 to 1984, the new organization reflects the diversity of the research now under way. The KSL is a modular laboratory, consisting of four collaborating yet distinct groups with different research themes:

- **The Heuristic Programming Project (HPP)**, Professor Edward A. Feigenbaum, scientific director—large, multi-use knowledge bases, blackboard systems, concurrent system architectures for AI, automated software design, expert systems for science and engineering. Executive director: Robert Engelmore. Research scientists: Harold Brown, Scott Clearwater, Bruce Delagi, Barbara Hayes-Roth, Hirotoshi Maegawa, H. Penny Nii, and Hiroshi Okuno.

- **The HELIX Group**, Professor Bruce G. Buchanan, scientific director—machine learning, transfer of expertise, and problem solving. Research scientists: James Brinkley, William J. Clancey, Craig Cornelius, Diana Forsythe, Barbara Hayes-Roth, Rich Keller, Catherine Manago.

- **The Medical Computer Science (MCS) Group**, Associate Professor Edward H. Shortliffe, scientific director (Department of Medicine with courtesy appointment in Computer Science)—fundamental research and advanced biomedical applications in the area of AI and decision sciences; includes the Medical Information Sciences (MIS) program. Assistant Professor: Mark A. Musen; Research scientists: Gregory F. Cooper, Lawrence M. Fagan (Associate Director).

- **The Symbolic Systems Resources Group (SSRG)**, Thomas C. Rindfleisch, scientific director (joint appointment Departments of Computer Science and Medicine)—research on and operation of distributed computing resources for AI research, including the SUMEX-AIM facility. Assistant director: William J. Yeager.

The KSL is guided by an Executive Committee consisting of the four sublaboratory directors. Tom Rindfleisch serves as overall KSL director.

This brochure summarizes the goals and methodology of the KSL, its research and academic programs, its achievements, and the research environment of the laboratory.

Basic Research Goals and Methodology

Throughout a 20-year history, the KSL and its predecessors, DENDRAL and HPP, have concentrated on research in expert systems—that is, systems using symbolic reasoning and problem-solving processes that are based on extensive domain-specific knowledge. The KSL's approach has been to focus on applications that are themselves significant real-world problems, in domains such as science, medicine, engineering, and education, and that also expose key, underlying AI research issues. For the KSL, AI is largely an empirical science. Research problems are explored, not by examining strictly theoretical questions, but by designing, building, and experimenting with programs that serve to test underlying theories.

The basic research issues at the core of the KSL's interdisciplinary approach center on the computer representation and use of large amounts of domain-specific knowledge, both factual and heuristic (or judgmental). These questions have guided our work since the 1960s and are now of central importance in all of AI research:

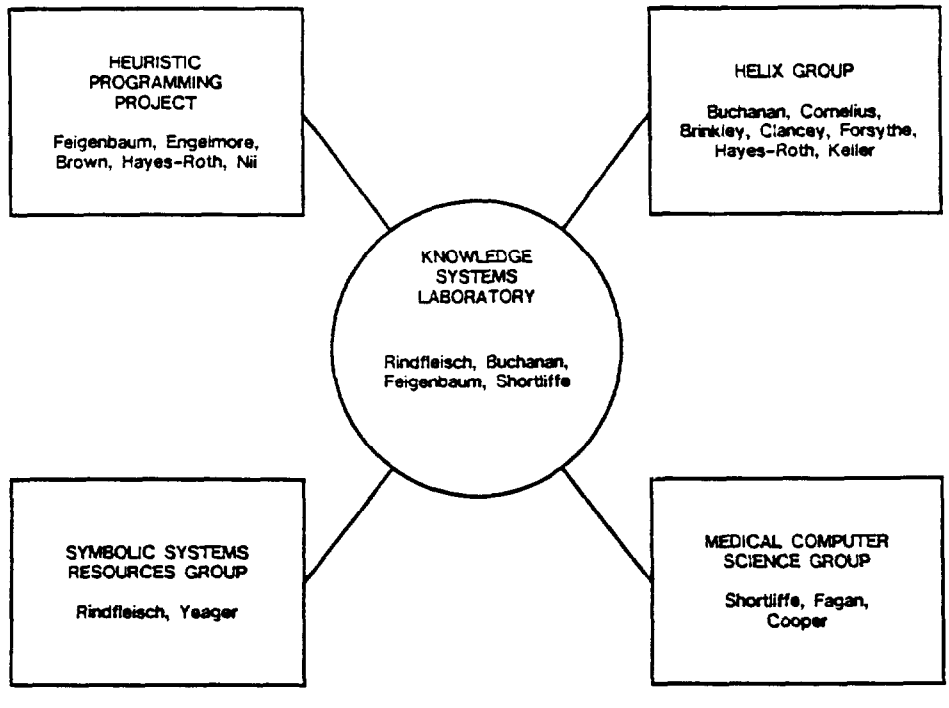
1. **Knowledge representation.** How can the knowledge necessary for complex problem solving be represented for its most effective use in automatic inference processes? Often, the knowledge obtained from experts is heuristic knowledge, gained from many years of experience. How can this knowledge, with its inherent vagueness and uncertainty, be represented and applied? How can knowledge be represented so that it can be used for many problem solving purposes?

2. **Knowledge acquisition.** How is knowledge acquired most efficiently—whether from human experts, from observed data, from experience, or by discovery? How can a program discover inconsistency and incompleteness in its knowledge base? How can knowledge be added without perturbing the established knowledge base unnecessarily?

3. **Use of knowledge.** By what inference methods can many sources of knowledge of diverse types be made to contribute jointly and efficiently toward solutions? How can knowledge be used intelligently, especially in systems with large knowledge bases, so that it is applied in an appropriate manner at the appropriate time?

4. **Explanation and tutoring.** How can the knowledge base and the line of reasoning used in solving a particular problem be explained to users? What constitutes a sufficient or an acceptable explanation for different classes of users?

5. **System tools and architectures.** What kinds of software tools and system architectures can be constructed to make it easier to implement expert programs with greater complexity and higher performance? What kinds of systems can serve as vehicles for the cumulation of knowledge of the field for the researchers?



Knowledge Systems Laboratory Organization

Current Research Projects

The following list of projects now under way within the four KSL research groups gives a brief summary of the major goals of each project and lists the personnel (staff and Ph.D. candidates) directly involved. More complete information on individual projects can be obtained from the person indicated as the project contact. Inquiries should be addressed in care of:

Knowledge Systems Laboratory
Department of Computer Science
Stanford University
701 Welch Road, Building C
Palo Alto, CA 94304
415-723-3444

The Heuristic Programming Project

- **Advanced Architectures Project**—Design a new generation of computer architectures to exploit concurrency in blackboard-based signal understanding systems.
  Personnel: Edward A. Feigenbaum (contact), Nelleke Aiello, Harold Brown, Bruce Delagi (DEC), Robert Engelmore, Hirotoshi Maegawa (Sony), Penny Nii, Sayuri Nishimura, Hiroshi Okuno (NTT), James Rice, Nakul Saraiya.

- **Blackboard Architecture Project**—Integrate current knowledge about blackboard framework problem-solving systems and develop a domain-independent model that includes knowledge-based control processes.
  Personnel: Barbara Hayes-Roth (contact), Micheal Hewett, Penny Nii.

- **Large Multi-use Knowledge Bases (LMKB)**—Develop a knowledge base of scientific and engineering facts, principles and methods, along with appropriate representations of the knowledge, for multiple uses, including diagnosis and monitoring, planning, configuration, and tutoring.
  Personnel: Edward Feigenbaum (contact), Richard Keller, Scott Clearwater (LANL), Robert Engelmore.

- **Automated Software Design**—Assist software designers in designing new program modules via intelligent selection and modification from a library of existing software modules.
  Personnel: Penny Nii(contact), Cordell Green (Kestrel Institute).

The HELIX Group

- **PROTEAN**—Study complex symbolic constraint-satisfaction problems in the blackboard framework with application to protein structure determination from nuclear magnetic resonance data.
  Personnel: Bruce Buchanan (contact), Oleg Jardetzky (Stanford Magnetic Resonance Laboratory), Russ Altman, Jim Brinkley, Enrico Carrara, Craig Cornelius, Bruce Duncan, Guido Haymann-Haber, Olivier Lichtarge.

- **NEOMYCIN/GUIDON2**—Develop knowledge representation and explanation capabilities for computer-aided teaching of diagnostic reasoning. This work is moving to the Xerox Institute for Research on Learning in Spring 1988.
  Personnel: Bill Clancey (contact), Stephen Barnhouse, Bob London, Steve Oliphant.

- **Knowledge Acquisition Studies**—Study the processes for transferring knowledge into a computer program, including learning by induction, analogy, watching, chunking, reading, and discovery.
  Personnel: Bruce Buchanan (contact), Martin Chavez, Tze-Pin Cheng, Diana Forsythe, Haym Hirsh, Richard Keller, Harold Lehmann, Eric Schoen, John Sullivan.

- **Financial Resources Management**—Develop a constraint-based expert system for financial resource planning.
  Personnel: Bruce Buchanan and Tom Rindfleisch (contacts), Craig Cornelius, Andy Gelman, Catherine Manago.

- **Large Multi-use Knowledge Bases (LMKB)**—See description under HPP.

The Medical Computer Science Group

- **ONCOCIN**—Develop knowledge-based systems for the administration of complex medical treatment protocols such as those encountered in cancer chemotherapy.
  Personnel: Ted Shortliffe (contact), Charlotte Jacobs (Oncology), Larry Fagan, David Combs, Robert Carlson, Christopher Lane, Curt Langlotz, Rick Lenon, Mark Musen, Janice Rohn, Samson Tu, Cliff Wulfman, Andrew Zelenetz.

- **OPAL/PROTEGE**—Develop graphics-based knowledge acquisition tools for clinical trials. OPAL developed out of the ONCOCIN project to provide a method for specifying cancer treatment experiments. The PROTEGE program is capable of creating OPAL-like knowledge acquisition tools for various areas of medicine.
  Personnel: Mark Musen (contact), Larry Fagan, Ted Shortliffe, David Combs, Eric Sherman.

- **Speech Input to Expert Systems**—Develop multi-modal interface to expert systems, concentrating on a connected speech input device. Primary application will be extension to the ONCOCIN graphical interface.
  Personnel: Larry Fagan (contact), Bonnie Webber (University of Pennsylvania), Ted Shortliffe, Ed Feigenbaum (HPP), Ellen Isaacs (Psycholinguistics), Clifford Wulfman.

- **Physician's Workstation**—Develop advanced integrated workstation suitable for providing decision support functions to clinicians in both inpatient and outpatient settings; initial work in the area of cardiovascular disease prevention, with an emphasis on the management of lipid disorders.
  Personnel: Ted Shortliffe (contact), John Schroeder (Cardiology), David Maron (Heart Disease Prevention Center), Jonathan King, Tom Rindfleisch, Don Rucker, Joan Walton.

- **Blackboard/Intensive Care Unit (BBICU)**—Interpret data from the intensive care unit and suggest therapy plans for patients with mechanical breathing support. Two aspects of the project are: (1) representing the structure and function of the body and (2) combining qualitative and quantitative reasoning techniques.
  Personnel: Larry Fagan (contact for qualitative/quantitative), Barbara Hayes-Roth (HPP - contact for structure/function), Adam Seiver (Palo Alto Veterans Hospital), Lewis Sheiner (University of California, San Francisco), Ingo Beinlich, Reed Hastings, Micheal Hewett, Noi Hewett, Michael Kahn (UCSF), Nick Parlante (Palo Alto VA Hospital), John Reed, George Thomsen, Rich Washington.

- **Probabilistic Expert Systems**—Develop pragmatic and theoretically sound methods for the acquisition and computation of probabilistic information within medical expert systems.
  Personnel: Greg Cooper (contact), Ted Shortliffe, David Heckerman, Eddie Herskovits, Eric Horvitz, Jaap Suermondt.

The Symbolic Systems Resources Group (SSRG)

- **SUMEX-AIM Resource**—Develop and operate a national computing resource for biomedical applications of artificial intelligence in medicine and for basic research in AI at KSL.
  Personnel: Tom Rindfleisch (contact), Rich Acuff, Mark Crispin, Frank Gilmurray, Michael Marria, Christopher Schmidt, Andrew Sweer, Bob Tucker, Nicholas Veizades, Bill Yeager.

- **AI Workstation and Network Systems**—Develop network-based computing environments for Lisp workstations including remote graphics and distributed computing.
  Personnel: SSRG staff

- **Financial Resources Management**—See description under HELIX.

Students and Special Degree Programs

Graduate students are an essential part of the research productivity of the KSL. Currently 36 students are working with our projects centered in Computer Science and another 21 students are working with the MCS/MIS programs in Medicine. Of the 36 working in Computer Science, 16 are working toward Ph.D. degrees, and 20 are working toward M.S. degrees. A number of these students are pursuing interdisciplinary programs and come from the Departments of Engineering, Mathematics, Education, and Medicine. Of the 21 working in Medicine, 15 are working toward Ph.D. degrees, and 6 are working toward M.S. degrees.

Because of the highly interdisciplinary and experimental nature of KSL research, two special degree programs have been established:

Medical Information Sciences (MIS)— an interdepartmental program approved by Stanford University in 1982. It offers instruction and research opportunities leading to the M.S. or Ph.D. degree in medical information sciences, with an emphasis on either medical computer science or medical decision science. The program, directed by Ted Shortliffe and co-directed by Larry Fagan, is formally administered by the School of Medicine, but the curriculum and degree requirements are coordinated with the Dean of Graduate Studies and the Graduate Studies Committee of the University. The program reflects our local interest in the interconnections between computer science, artificial intelligence, and medical problems. Emphasis is placed on providing trainees with a broad conceptual overview of the field and with an ability to create new theoretical and practical innovations of clinical relevance.

Master of Science in Computer Science: Artificial Intelligence (MS:AI)— a terminal professional degree offered for students who wish to develop a competence in the design of substantial knowledge-based AI applications but who do not intend to obtain a Ph.D. degree. The MS:AI program is administered by the Committee for Applied Artificial Intelligence, composed of faculty and research staff of the Computer Science Department. Normally, students spend two years in the program with their time divided equally between course work and research. In the first year, the emphasis is on acquiring fundamental concepts and tools through course work and project involvement. During the second year, students implement and document a substantial AI application project.

Academic and Research Achievements

The primary products of our research are scientific publications on the basic research issues that motivate our work, computer software in the form of the expert systems and AI architectures we develop, and the students we graduate who continue AI research in other academic and industrial laboratories.

The KSL has averaged publishing more than 45 research papers per year in the AI literature, including journal articles, theses, proceedings articles, and working papers.* In addition, many talks and invited lectures are given annually. In the past few years, 11 major books have been published by KSL faculty, staff, and former students, and several more are in progress. Those recently published include:

- *Heuristic Reasoning about Uncertainty: An AI Approach,* Cohen, Pitman, 1985.
- *Readings in Medical Artificial Intelligence: The First Decade,* Clancey and Shortliffe, Addison-Wesley, 1984.
- *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project,* Buchanan and Shortliffe, Addison-Wesley, 1984.
- *The Fifth Generation: Artificial Intelligence and Japan's Computer Challenge to the World,* Feigenbaum and McCorduck, Addison-Wesley, 1983.
- *Building Expert Systems,* F. Hayes-Roth, Waterman, and Lenat, eds., Addison-Wesley, 1983.
- *System Aids in Constructing Consultation Programs: EMYCIN,* van Melle, UMI Research Press, 1982.
- *Knowledge-Based Systems in Artificial Intelligence: AM and TEIRESIAS,* Davis and Lenat, McGraw-Hill, 1982.
- *The Handbook of Artificial Intelligence,* Volume I, Barr and Feigenbaum, eds., 1981; Volume II, Barr and Feigenbaum, eds., 1982; Volume III, Cohen and Feigenbaum, eds., 1982; Kaufmann.
- *Applications of Artificial Intelligence for Organic Chemistry: The DENDRAL Project,* Lindsay, Buchanan, Feigenbaum, and Lederberg, McGraw-Hill, 1980.

Our laboratory has pioneered in the development and application of AI methods to produce high-performance knowledge-based programs. Programs have been developed in such diverse fields as analytical chemistry (DENDRAL), infectious disease diagnosis and treatment (MYCIN), cancer chemotherapy management (ONCOCIN), pulmonary function evaluation (PUFF), VLSI design (KBVLSI/PALLADIO), molecular biology (MOLGEN), and parallel machine architecture simulation (CARE). Some of our systems and tools (e.g., UNITS, EMYCIN, and AGE) are now also being adapted for commercial development and use in the AI industry.

Following our lead in work on biomedical applications of AI and the development of the SUMEX-AIM computing resource, a nationally recognized community of academic projects on AI in medicine has grown up.

Central to all KSL research are our faculty, staff, and students. These people have been recognized internationally for the quality of their work and for their continuing contributions to the field. KSL members participate extensively in professional organizations, government advisory committees, and journal editorial boards. They have held managerial posts and conference chairmanships in both the American Association for Artificial Intelligence (AAAI) and the International Joint Conference on Artificial Intelligence (IJCAI).

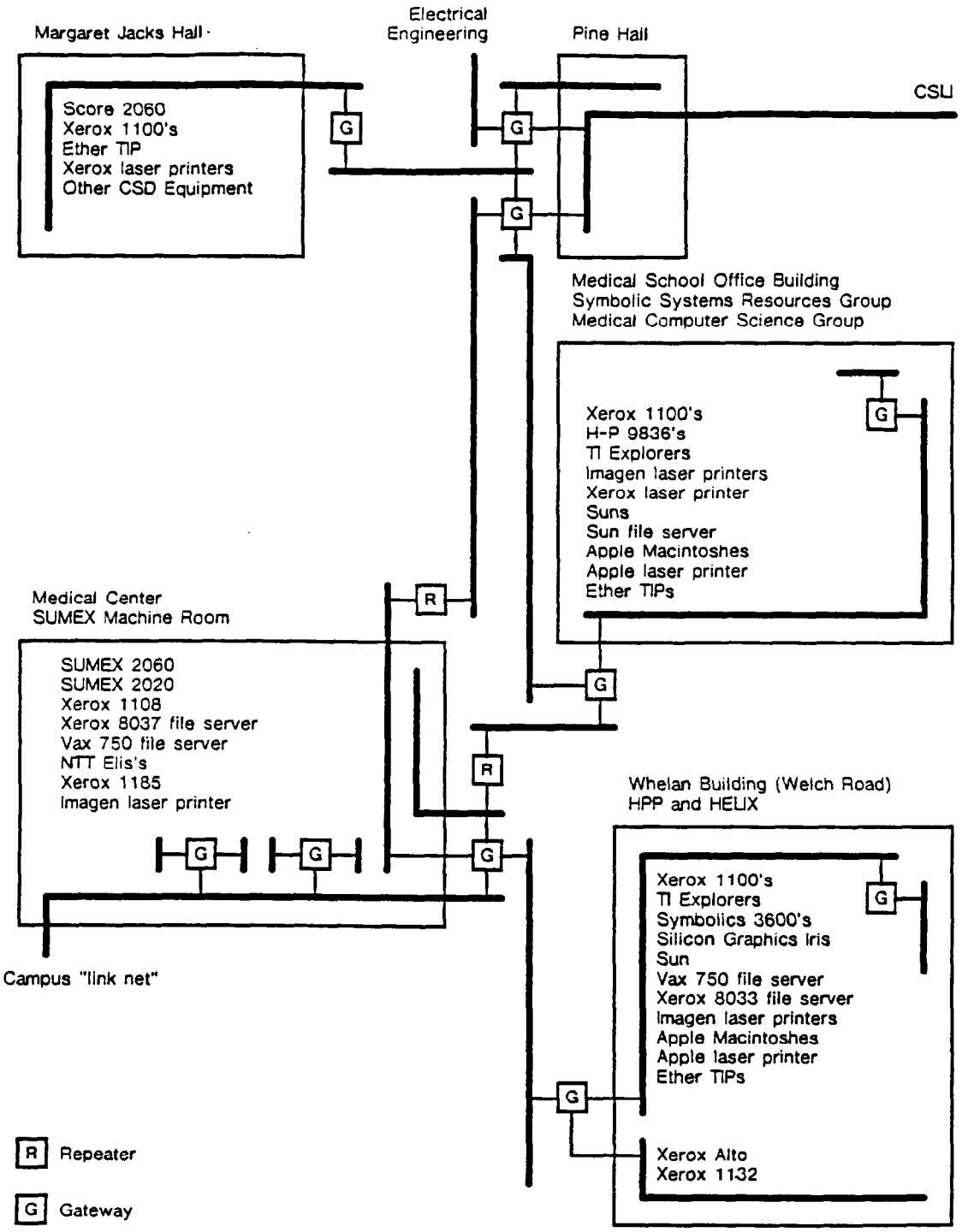
* Copies of individual KSL publications may be obtained through the Stanford Department of Computer Science publications office. The full collection of KSL reports is being published in microfiche by COMTEX Scientific Corporation.

Several KSL faculty and former students have received significant honors. In 1976, Ted Shortliffe received the Association of Computing Machinery Grace Murray Hopper award. In 1977, Doug Lenat was given the IJCAI Computers and Thought award, and in 1978, Ed Feigenbaum received the National Computer Conference Most Outstanding Technical Contribution award. In 1979 and 1981, Ted Shortliffe's book *Computer-Based Medical Consultation: MYCIN* was identified as the most frequently cited work in the IJCAI proceedings. In 1982, Doug Lenat won the Tioga prize for the best AAAI conference paper while Mike Genesereth received honorable mention. In 1983, Ted Shortliffe was named a Kaiser Foundation faculty scholar, and Tom Mitchell received the IJCAI Computers and Thought award. In 1984, Ed Feigenbaum was elected a fellow of the American Association for the Advancement of Science (AAAS), and he and Ted Shortliffe were elected fellows of the American College of Medical Informatics. In 1986, Ed Feigenbaum was elected to the National Academy of Engineering and in 1987, Ted Shortliffe was elected to the Institute of Medicine of the National Academy of Sciences.

KSL Research Environment

Funding—The KSL is supported solely by sponsored research and gift funds. We have had funding from many sources, including DARPA, NIH/NLM, ONR, NSF, NASA, and private foundations and industry. Of these, DARPA and NIH have been the most substantial and long-standing sources of support. All, however, have made complementary contributions to establishing an effective overall research environment that fosters interchanges at the intellectual and software levels and that provides the necessary physical computing resources for our work.

Computing Resources—Under the Symbolic Systems Resources Group, the KSL develops and operates its own computing resources tailored to the needs of its individual research projects. Current computing resources are a networked mixture of mainframe host computers, Lisp workstations, and network utility servers, reflecting the evolving hardware technology available for AI research. Our mainframe host is currently a DEC 2060 running TOPS-20 (this is the core of the national SUMEX biomedical computing resource). Its network service functions will be replaced shortly by a SUN-4 system running UNIX. Its routine computing functions (electronic mail, text processing, and information retrieval) will be replaced by distributed user workstations. Our Lisp workstations include 35 Xerox 1100-series machines, 20 Texas Instruments Explorers, 6 Symbolics 3600-series machines, 3 SUN 3/75 workstations, and 5 Hewlett-Packard 9836 machines. We are in the process of acquiring a significant number of Apple Macintosh II workstations for routine computing support and many of these will also be configured to run Lisp programs. Network printing, file, gateway, and terminal interface services are provided by dedicated machines including 2 VAX 11/750's, a SUN 3/180, and numerous dedicated microprocessor systems. These facilities are integrated with other computer science resources at Stanford through an extensive Ethernet and to external resources through the ARPANET and TELENET. Funding for these resources comes principally from DARPA and NIH and hardware vendor gifts.



SUMEX-AIM System and Local Area Network

Appendix B

Lisp Performance Studies

Performance of Two Common Lisp Programs on Various Workstation Systems

by Richard Acuff
Knowledge Systems Laboratory
Stanford University

*** DRAFT ***

1 - Introduction

In order to assist us in understanding performance of Lisp systems, we have undertaken an informal survey of Common Lisp environments using two KSL software packages. The data collection is close to complete but there has been very little data analysis. Thus the data is included here with very little in the way of observations or conclusions.

In this survey we have focused on execution speed which has long been a differentiator among computer systems. The first comparison of two systems solving the same problem (benchmarking) was probably done shortly after the creation of the second computer, and benchmarking has been a primary differentiator among computers systems ever since. However, execution speed benchmarks are only one aspect of the systems, especially Lisp systems. Issues like programming and usage environments, compatibility with other systems, ability to handle "large" problems, and cost must also be considered.

The test software we used was SOAR and the BB1 blackboard core. Both systems were chosen primarily because they are implemented in pure Common Lisp, making them extremely portable. Both are systems in daily use in the KSL and represent two distinct research directions. SOAR is a heuristic search based general problem solving architecture developed by Paul Rosenbloom and BB1 is a blackboard problem solving architecture developed by Barbara Hayes-Roth. Neither of these systems is an intensive user of numeric computation. These systems were initially developed in environments other than those tested and no attempt was made to optimize their performance for any of these tests.

All runs of SOAR were done solving an eight-puzzle problem in one of three modes:

1. Mode "1,3" just solves the problem.
2. Mode "1,1" solves the problem while learning how to better solve it (this mode takes the most time).
3. Mode "3,3" solves the problem after learning (this mode takes the least time).

SOAR's source code consisted of a single 280k character file, plus two small files containing the "rules" for the eight-puzzle problem: DEFAULT.SOAR at 24k characters and EIGHT.SOAR at 10k characters. The runs and compilation were done in separate instantiations of the Lisp environment.

All runs of BB1 went through three cycles of adding 10 items to the blackboard, accessing those 10 items, and then deleteing them. All references to BB1 in this document refer only to the "core" blackboard parts of the system and does not include any other layers of the problem solving architecture, or the user interface. The BB1 source code used for the testing is spread over 10 files containing 295k characters. Compilation, loading, and execution were all done in a single instantiation of the Lisp environment.

2 - Systems Under Test

The systems to be tested were chosen based on their availability to the testers as well as suspected potential usefulness in future programming efforts. Since we were interested in "real world" results, we ran the tests on each machine in what seemed to be its standard operating mode. In particular, if there are typically "background" activities going on during normal operation, then those were allowed to continue during the taking of these measurements. If code is typically executed from within an editor or other special context, then that was done. No special process priority altering or other attempt to optimize the execution was made unless noted in the description of the systems.

It is worth noting that on almost all of the systems tested, virtual memory paging was a neglibible part of the overall run time. Nor was it a significant factor during compilation.

In the following descriptions "Code" refers to a short name used to indicate the systems under test. Usually it is the model of the machine except where there is more than one Lisp for a machine (as in the case of the Sun 3/75) in which case a letter is prefixed to indicate the Lisp being used. "Timing Template" indicates how the information reported by the TIME function was recorded. "Elapsed" indicates the total elapsed time, "run" indicates CPU time used, "gc" indicates time spent in garbage collection, "user" and "system" distinguish between user mode and kernel mode time, and "paging" indicates time waiting for virtual memory disk operations. Though all of this information was recorded we have not reproduced it in this document.

Code: 3/260
Computer Type: Sun 3/260
Operating System: Sun OS 3.4
Lisp: Lucid 2.0
Disk Configuration: 280MB
Swapping Size: 60MB
Memory Configuration: 8MB
Display Configuration: Color in mono mode
Other Configuration:
Special Comments: used :EXPAND 130 :GROWTH-RATE 130
Timing Template: elapsed (user-run + system-run)

Code: 3/60
Computer Type: Sun 3/60
Operating System: Sun OS 3.4
Lisp: Lucid 2.1
Disk Configuration: SCSI 141MB
Swapping Size: unknown
Memory Configuration: 24MB
Display Configuration: Hi Res Color in mono mode
Other Configuration:
Special Comments:

Appendix B

Lisp Performance Studies

Performance of Two Common Lisp Programs on Various Workstation Systems

by Richard Acuff
Knowledge Systems Laboratory
Stanford University

*** DRAFT ***

1 - Introduction

In order to assist us in understanding performance of Lisp systems, we have undertaken an informal survey of Common Lisp environments using two KSL software packages. The data collection is close to complete but there has been very little data analysis. Thus the data is included here with very little in the way of observations or conclusions.

In this survey we have focused on execution speed which has long been a differentiator among computer systems. The first comparison of two systems solving the same problem (benchmarking) was probably done shortly after the creation of the second computer, and benchmarking has been a primary differentiator among computers systems ever since. However, execution speed benchmarks are only one aspect of the systems, especially Lisp systems. Issues like programming and usage environments, compatibility with other systems, ability to handle "large" problems, and cost must also be considered.

The test software we used was SOAR and the BB1 blackboard core. Both systems were chosen primarily because they are implemented in pure Common Lisp, making them extremely portable. Both are systems in daily use in the KSL and represent two distinct research directions. SOAR is a heuristic search based general problem solving architecture developed by Paul Rosenbloom and BB1 is a blackboard problem solving architecture developed by Barbara Hayes-Roth. Neither of these systems is an intensive user of numeric computation. These systems were initially developed in environments other than those tested and no attempt was made to optimize their performance for any of these tests.

All runs of SOAR were done solving an eight-puzzle problem in one of three modes:

1. Mode "1,3" just solves the problem.
2. Mode "1,1" solves the problem while learning how to better solve it (this mode takes the most time).
3. Mode "3,3" solves the problem after learning (this mode takes the least time).

SOAR's source code consisted of a single 280k character file, plus two small files containing the "rules" for the eight-puzzle problem: DEFAULT.SOAR at 24k characters and EIGHT.SOAR at 10k characters. The runs and compilation were done in separate instantiations of the Lisp environment.

All runs of BB1 went through three cycles of adding 10 items to the blackboard, accessing those 10 items, and then deleteing them. All references to BB1 in this document refer only to the "core" blackboard parts of the system and does not include any other layers of the problem solving architecture, or the user interface. The BB1 source code used for the testing is spread over 10 files containing 295k characters. Compilation, loading, and execution were all done in a single instantiation of the Lisp environment.

2 - Systems Under Test

The systems to be tested were chosen based on their availability to the testers as well as suspected potential usefulness in future programming efforts. Since we were interested in "real world" results, we ran the tests on each machine in what seemed to be its standard operating mode. In particular, if there are typically "background" activities going on during normal operation, then those were allowed to continue during the taking of these measurements. If code is typically executed from within an editor or other special context, then that was done. No special process priority altering or other attempt to optimize the execution was made unless noted in the description of the systems.

It is worth noting that on almost all of the systems tested, virtual memory paging was a neglibible part of the overall run time. Nor was it a significant factor during compilation.

In the following descriptions "Code" refers to a short name used to indicate the systems under test. Usually it is the model of the machine except where there is more than one Lisp for a machine (as in the case of the Sun 3/75) in which case a letter is prefixed to indicate the Lisp being used. "Timing Template" indicates how the information reported by the TIME function was recorded. "Elapsed" indicates the total elapsed time, "run" indicates CPU time used, "gc" indicates time spent in garbage collection, "user" and "system" distinguish between user mode and kernel mode time, and "paging" indicates time waiting for virtual memory disk operations. Though all of this information was recorded we have not reproduced it in this document.

Code: 3/260
Computer Type: Sun 3/260
Operating System: Sun OS 3.4
Lisp: Lucid 2.0
Disk Configuration: 280MB
Swapping Size: 60MB
Memory Configuration: 8MB
Display Configuration: Color in mono mode
Other Configuration:
Special Comments: used :EXPAND 130 :GROWTH-RATE 130
Timing Template: elapsed (user-run + system-run)

Code: 3/60
Computer Type: Sun 3/60
Operating System: Sun OS 3.4
Lisp: Lucid 2.1
Disk Configuration: SCSI 141MB
Swapping Size: unknown
Memory Configuration: 24MB
Display Configuration: Hi Res Color in mono mode
Other Configuration:
Special Comments:

Timing Template: elapsed (user-run + system-run)

Code: 386
Computer Type: Compaq 386 (20Mhz 386)
Operating System: 386/IX 5.3 rev level 1.01 (unix)
Lisp: Lucid 2.0
Disk Configuration: 134MB ESDI
Swapping Size: unknown
Memory Configuration: 10MB; 32kB 20ns cache
Display Configuration: terminal
Other Configuration: none
Special Comments: none
Timing Template: elapsed (run)

Code: 386T
Computer Type: Compaq 386 portable (Toaster)
Operating System: 386/IX 5.3 rev level 1.01 (unix)
Lisp: Lucid 2.0
Disk Configuration: 40MB
Swapping Size: unknown
Memory Configuration: 10MB; no cache
Display Configuration: tiny LCD
Other Configuration: tiny display
Special Comments: portable versio of "386" above
Timing Template: elapsed (run)

Code: 4/260
Computer Type: Sun 4/280
Operating System: SunOS 3.2 Gamma
Lisp: Lucid 2.1
Disk Configuration: unknown
Swapping Size: unknown
Memory Configuration: 32MB
Display Configuration: Hi Res color in mono
Other Configuration:
Special Comments: used :EXPAND 130 :GROWTH-RATE 130
Timing Template: elapsed (user-run + system-run)

Code: 4/280
Computer Type: Sun 4/280
Operating System: SunOS 3.2 Gamma
Lisp: Lucid 2.1 beta
Disk Configuration: 417 (Eagle)
Swapping Size: 60MB
Memory Configuration: 8MB
Display Configuration: Hi Res mono
Other Configuration:
Special Comments:
Timing Template: elapsed (user-run + system-run)

Code: DEC-II
Computer Type: DEC MicroVax II/GPX
Operating System: VMS
Lisp: VaxLisp
Disk Configuration: 2 x 159MB
Swapping Size: 3k pg page, 8k pg swap

Memory Configuration: 16MB
Display Configuration: GPX
Other Configuration:
Special Comments:
Timing Template: elapsed - gc-elapsed (run - gc-run)

Code: DEC-III
Computer Type: DEC MicroVax III (3500)
Operating System: VMS
Lisp: VaxLisp
Disk Configuration: (RD53)
Swapping Size: unkown
Memory Configuration: 16MB
Display Configuration:
Other Configuration:
Special Comments:
Timing Template: elapsed - gc-elapsed (run - gc-run)

Code: E-3/75
Computer Type: Sun 3/75
Operating System: SunOS 3.1
Lisp: Franz Extended Common Lisp 2.0
Disk Configuration: 70MB SCSI
Swapping Size: 50MB local
Memory Configuration: 28MB
Display Configuration: standard resolution mono
Other Configuration: Files on Sun 3/180 NFS server
Special Comments: Under suntools
Timing Template: elapsed (run + gc)

Code: EXP1
Computer Type: Texas Instruments Explorer I
Operating System: Explorer Lisp Release 3.0+
Lisp: Explorer Lisp Release 3.0+
Disk Configuration: 2 x 140MB SCSI
Swapping Size: 80MB
Memory Configuration: 8MB
Display Configuration: 1024 x 768 mono
Other Configuration:
Special Comments: TGC (incremental generation scavenging GC) on unless otherwise noted
Timing Template: elapsed - paging

Code: EXP2
Computer Type: Texas Instruments Explorer II
Operating System: Explorer Lisp Release 3.0+
Lisp: Explorer Lisp Release 3.0+
Disk Configuration: 2 x 140MB SCSI
Swapping Size: 80MB
Memory Configuration: 16MB
Display Configuration: 1024 x 768 mono
Other Configuration:
Special Comments: TGC (incremental generation scavenging GC) on unless otherwise noted
Timing Template: elapsed - paging

Code: HP
Computer Type: Hewlett Packard 9000/350
Operating System: Unix
Lisp: HP Lisp 1.0
Disk Configuration: 130MB (7958)
Swapping Size: unknown
Memory Configuration: 16MB
Display Configuration: color
Other Configuration: under gnuemacs
Special Comments:
Timing Template: elapsed - run

Code: K-3/75
Computer Type: Sun 3/75
Operating System: SunOS 3.1
Lisp: Kyoto Common Lisp "September 16, 1986"
Disk Configuration: 70MB SCSI
Swapping Size: 50MB local
Memory Configuration: 28MB
Display Configuration: standard resolution mono
Other Configuration: Files on Sun 3/180 NFS server
Special Comments: Under suntools
Timing Template: elapsed - run

Code: L-3/75
Computer Type: Sun 3/75
Operating System: SunOS 3.1
Lisp: Lucid 2.0
Disk Configuration: 70MB SCSI
Swapping Size: 50MB local
Memory Configuration: 28MB
Display Configuration: standard resolution mono
Other Configuration: Files on Sun 3/180 NFS server
Special Comments: used :EXPAND 90 :GROWTH-RATE 90
Timing Template: elapsed (user-run + system-run)

Code: mX
Computer Type: Texas Instruments microExplorer
Operating System: Explorer Lisp 4.0 beta
Lisp: Explorer Lisp 4.0 beta
Disk Configuration: 100MB Rodime
Swapping Size: 60MB
Memory Configuration: 12MB mX processor; 2MB Mac II
Display Configuration: 19" (1024 x 768) Moniterm Viking
Other Configuration: Apple EtherTalk
Special Comments:
Timing Template: elapsed - paging

Code: RT
Computer Type: IBM RT/APC
Operating System: AIX 2.1.2 (unix)
Lisp: 2.0.5 (Lucid 1.01)
Disk Configuration: "Fast" EESDI controller; 3 x 70MB
Swapping Size: 80k x 512kB blocks (40,960MB)
Memory Configuration: 16MB of "fast" memory
Display Configuration: Moniterm 1024 x 768 mono

Other Configuration: AFT floating point unit; GSL windows
Special Comments: Used :EXPAND 69 to get 6MB semispace; This should be the fastest RT version now available
Timing Template: elapsed (user-run + system-run)

Code: Sym
Computer Type: Symbolics 3645
Operating System: Symbolics Release 6.1
Lisp: Symbolics Release 6.1
Disk Configuration: 368MB
Swapping Size: 200MB
Memory Configuration:
Display Configuration: 8MB
Other Configuration: FPA, no color
Special Comments: EGC on
Timing Template: elapsed - paging

Code: XCL
Computer Type: Xerox 1186
Operating System: Xerox Lisp, Lyric release
Lisp: Xerox Lisp, Lyric release
Disk Configuration: 40MB
Swapping Size: 16MB
Memory Configuration: 3.5MB
Display Configuration: 19" mono
Other Configuration:
Special Comments:
Timing Template: elapsed - gc - paging

3 - Compilation and Execution

For both BB1 and SOAR the time taken to compile the system and make a standard run was measured. Here are those results (all numbers are seconds):

	BB1		SOAR	
Code	Compile	Run	Compile	Run
3/260	540	62	687	171
3/60	551	73	569	218
386	355	47	386	142
386T	416	54	479	175
4/260	324	56	307	70
4/280	482	34	523	105
DEC-II	1774	207	1227	1908
DEC-III	633	63	423	476
E-3/75	444	211	450	500
Exp1	327	87	520	400
Exp2	96	29	162	146
HP	235	115	237	229
L-3/75	919	90	1365	756
K-3/75	1234	96	1040	297
MacII	349	254	Not available[1]	
mX	242	31	242	219
RT	586	75	574	206
Sym	257	111	252	210
XCL	1927	559	1800	1613

[1] We were not able to get SOAR to run properly in Allegro Common Lisp 1.0 or 1.1.

For convenience of viewing, these numbers are graphed in Figures 20 and 21. The system types are sorted into order of best run time for SOAR and BB1 separately to facilitate comparative observation.

4 - Effect of Compiler Optimize Settings on BB1

We were also interested in the effect of two of Common Lisp's compiler optimizer settings, SPEED and SAFETY. These switches have four settings, 0 through 3, with 0 being the highest priority. Thus settings of SAFETY 0 and SPEED 3 should allow the compiler to produce the fastest code, while SAFETY 3 and SPEED 0 would result in conservative code, perhaps with more type checking, etc. We comipiled and ran BB1 with 4 settings of these switches:

1. The system default
2. (PROCLAIM '(OPTIMIZE (SAFETY 0) (SPEED 3)))
3. (PROCLAIM '(OPTIMIZE (SAFETY 3) (SPEED 0)))
4. (PROCLAIM '(OPTIMIZE (SAFETY 2) (SPEED 3)))

Figures 22 and 23 show the effect of these settings on the compilation and run times of the BB1 test. Here are the numbers:

	Default		Safe 0, Spd 3		Safe 3, Spd 0		Safe 2, Spd 3	
Code	Comp	Run	Comp	Run	Comp	Run	Comp	Run
3/260	540	62	524	62	532	69	527	62
3/60	551	73	537	72	444	76	540	72
386	355	47	332	47	271	52	339	47
386T	416	54	408	54	341	60	410	54
4/260	324	56	318	46	229	47	309	46
4/280	482	34	483	34	385	48	492	34
DEC-II	1774	207	1987	206	2245	231	3094	236
DEC-III	633	63	635	60	732	71	990	70
E-3/75	444	211	403	215	469	206	443	206
Exp1	327	87	318	87	314	90	357	83
Exp2	96	29	117	25	111	28	121	26
HP	235	115	250	113	235	141	247	118
K-3/75	1234	96	1815	165	1379	147	1171	88
L-3/75	919	90	1056	90	1054	127	910	90
MacII	349	254	365	258	354	261	363	259
mX	242	34	249	28	232	32	239	35
RT	586	75	587	76	508	77	595	75
Sym	257	111	281	109	256	110	262	111
XCL	1927	559	2022	543	2230	559	2020	556

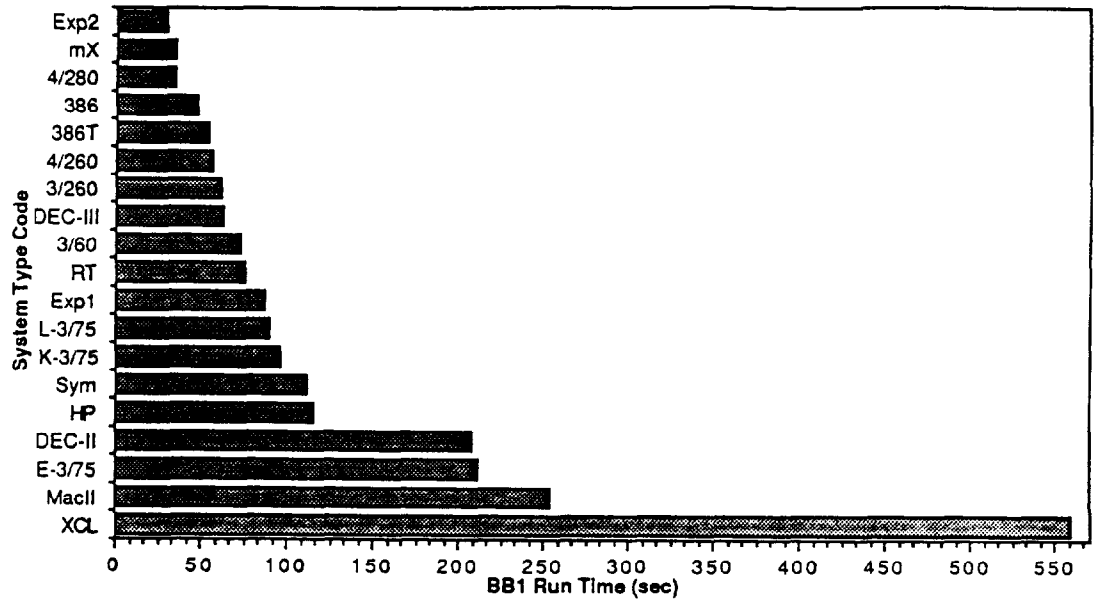
5 - Effect of Output Reduction on SOAR

We had previously noted that some systems were able to run the eight-puzzle benchmark much faster when the volumous typeout produced was reduced. Figure 24 shows the difference between run times of the 1,3 mode solution of the eight puzzle with full tracing versus no tracing. The numerical data follows:

Code	Normal	Reduced
3/260	49	33
3/60	66	38
386	41	27
386T	52	31
4/260	23	13
4/280	35	15
DEC-II	351	283
DEC-III	95	76
E-3/75	124	109
Exp1	90	63
Exp2	44	21
HP	61	51
K-3/75	186	136
L-3/75	82	67
mX	48	38
RT	61	36
Sym	55	40
XCL	473	390

6 - Future Work

Now that this data is collected it is our intention to write it up in a technical report, including comparisons with the Gabriel Benchmarks and remarks on certain surprising facts that this work has turned up. This report will be made widely available to the AIM community to assist future descisions in the use of Lisp systems.



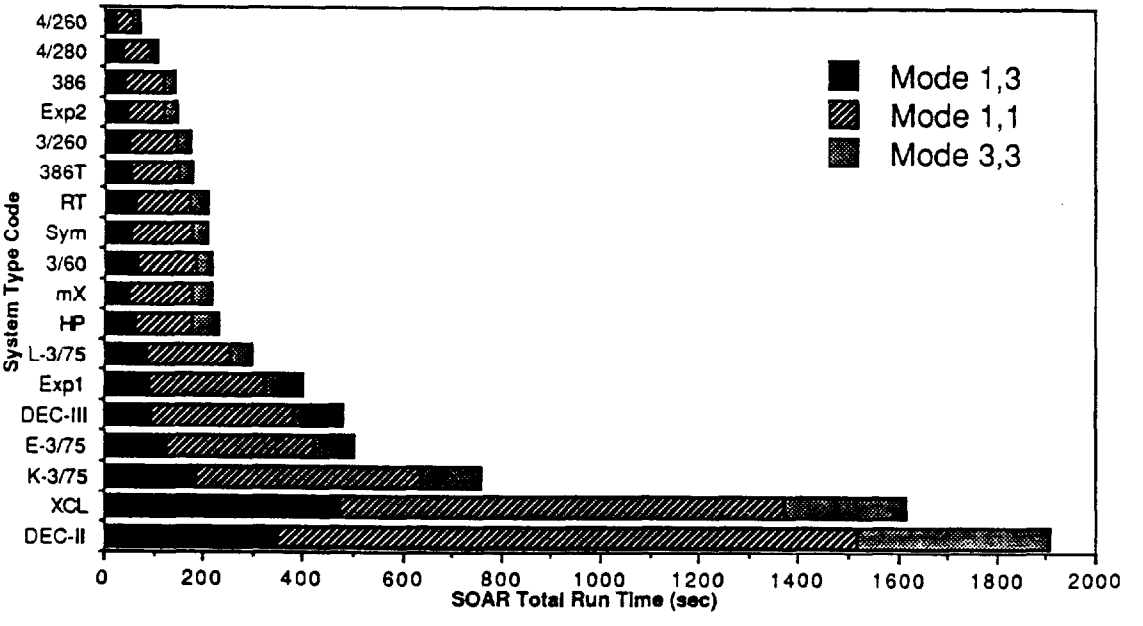


Figure 20: Run Times for BB1 and SOAR

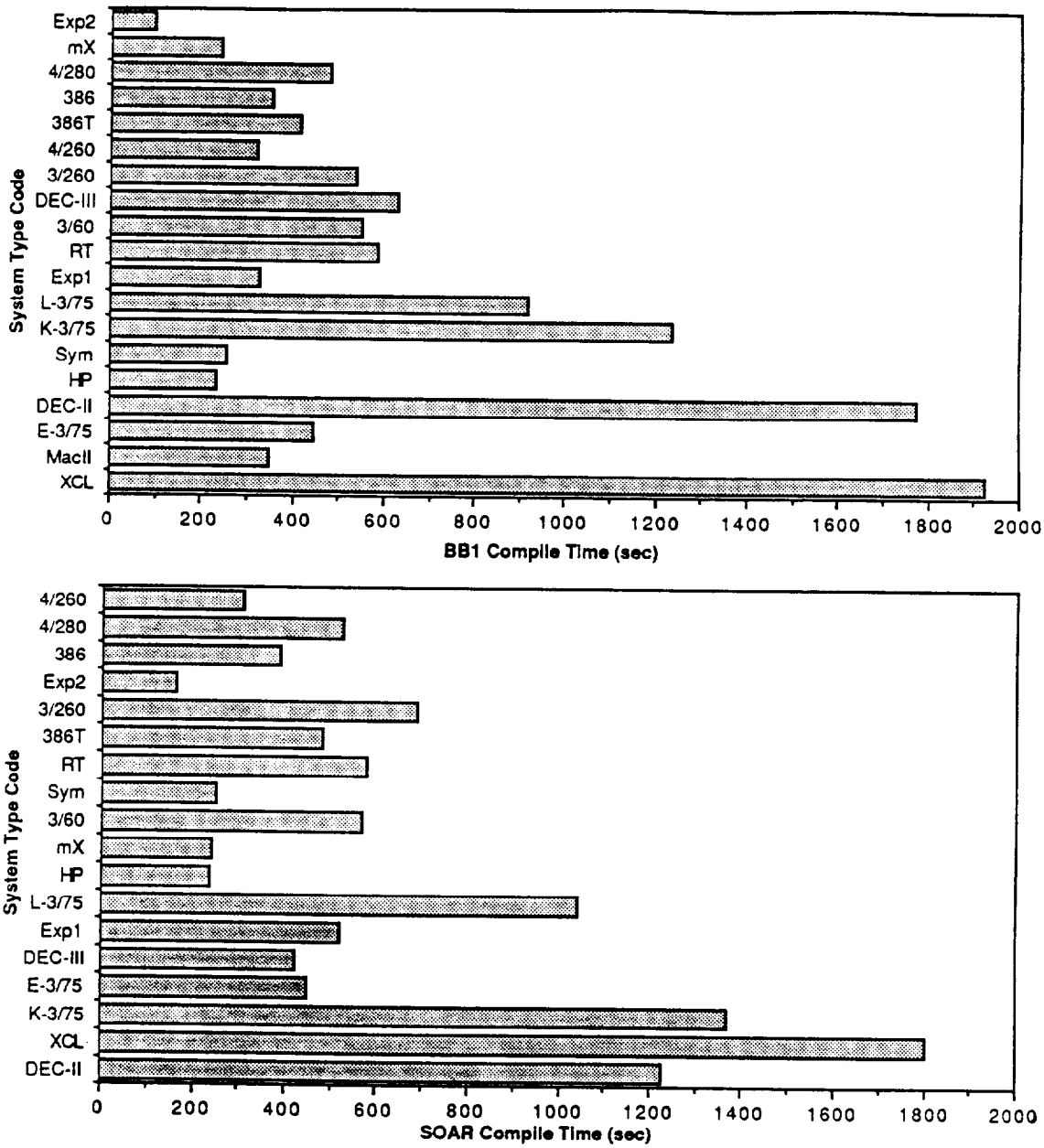


Figure 21: Compilation Times for BB1 and SOAR

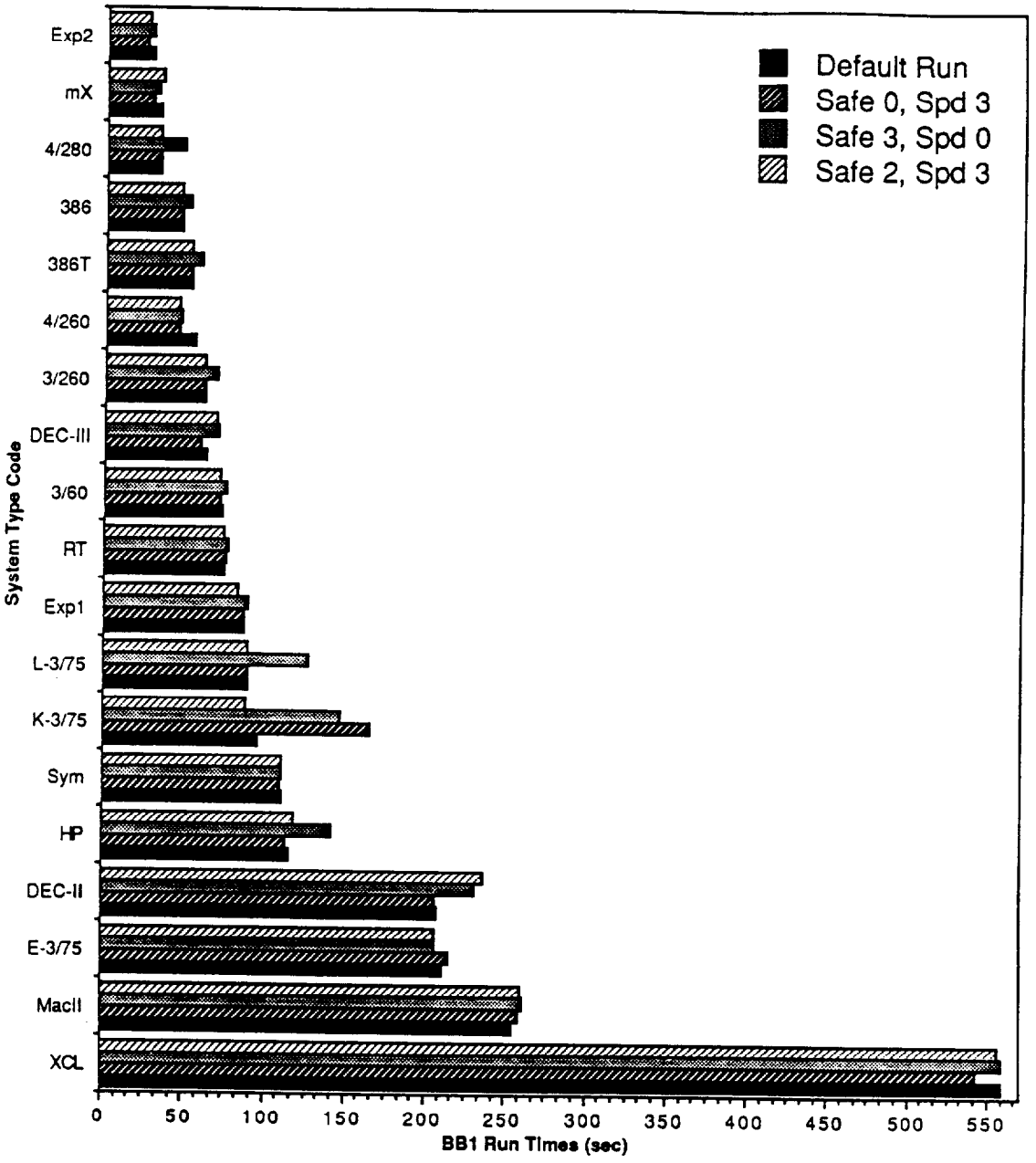


Figure 22: Run Times for BB1 under Various Compiler Settings

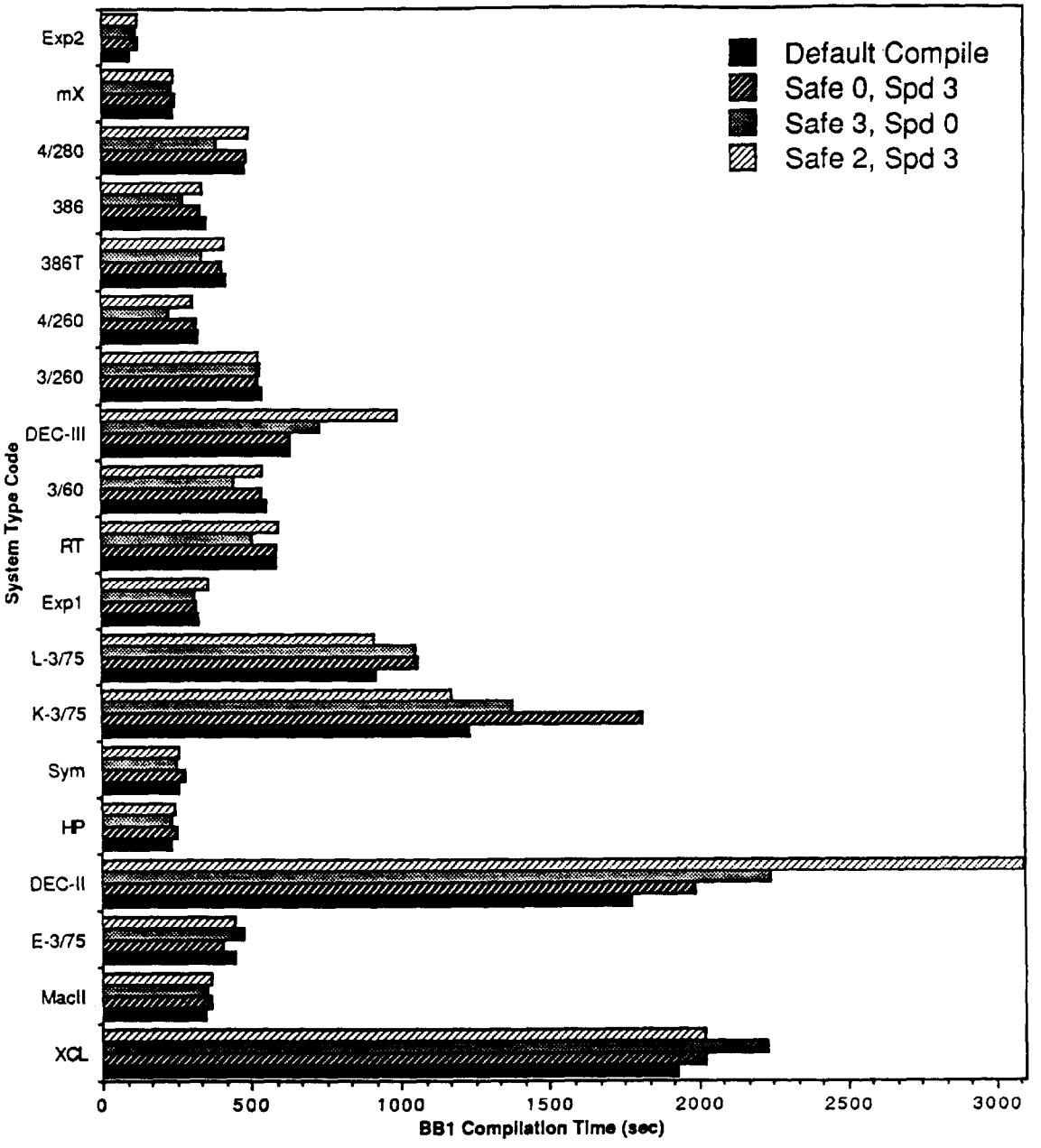


Figure 23: Compilation Times for BB1 under Various Compiler Settings

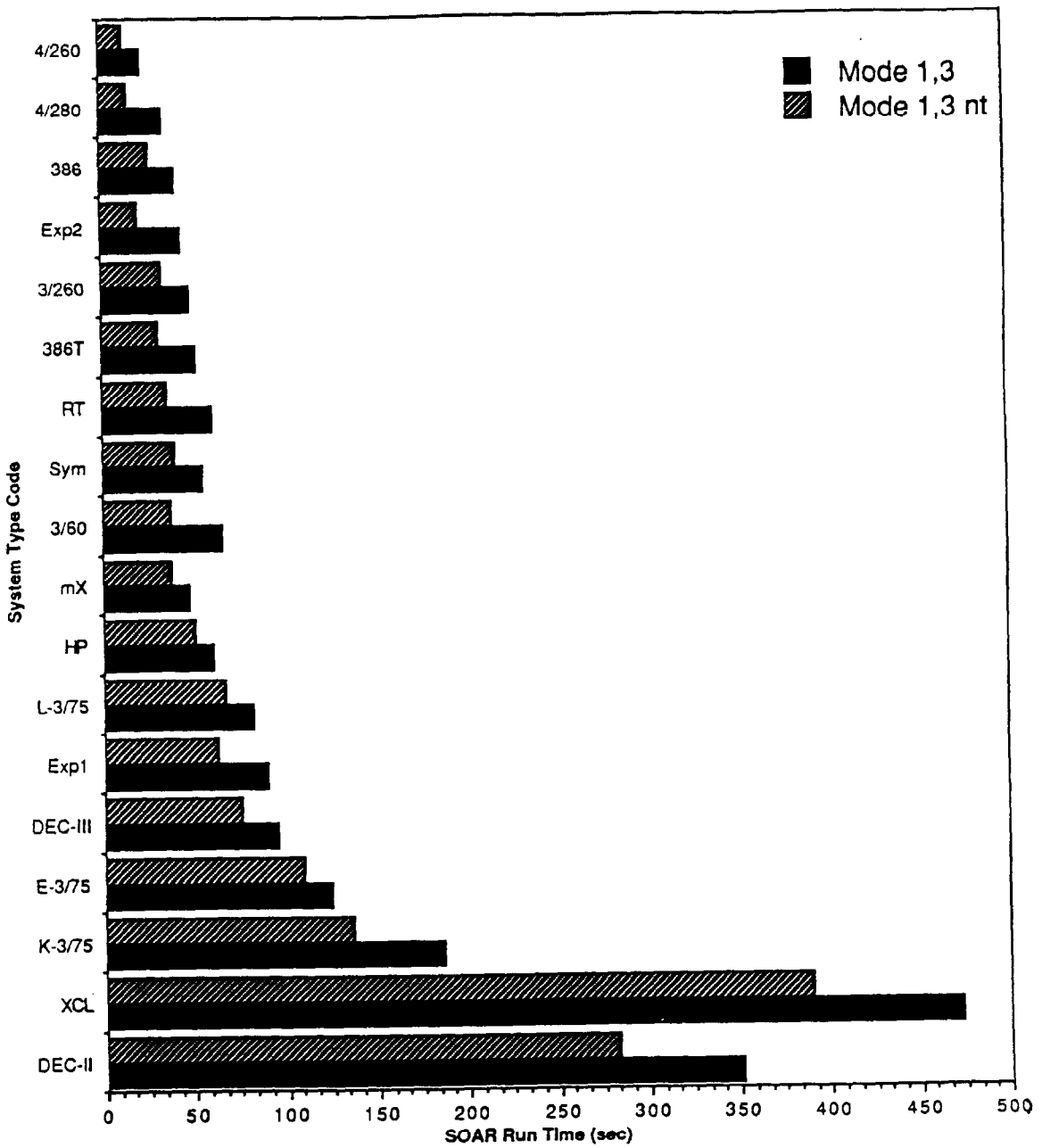


Figure 24: Differences Between Normal and Reduced Ouput SOAR Run